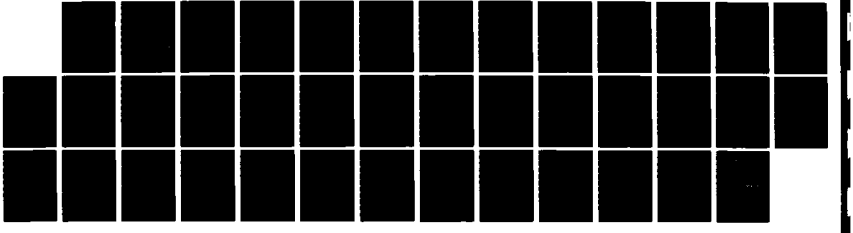
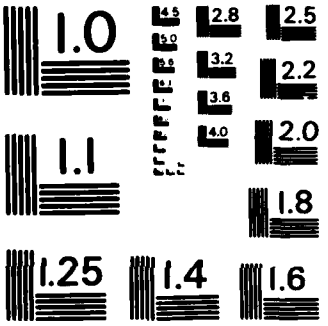
AD-A162 224 MASKDOC(U) BEDFORD RESEARCH ASSOCIATES MA J STEVENS 1/1
AUG 85 SCIENTIFIC-1 AFGL-TR-85-0169 F19628-83-C-0090

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

(12)

AFGL-TR-85-0169

MASK.DOC

J. Stevens

Bedford Research Associates
4 DeAngelo Drive
Bedford, MA 01730

AD-A162 224

Scientific Report No. 1

August 1985

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
DEC 9 1985
B

AIR FORCE GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731

DTIC FILE COPY

85 12 -9 073

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved for publication.

P Tsipouras

PAUL TSIPOURAS
Contract Manager

FOR THE COMMANDER

John E Holdner

LT COL J.E. Holdner, Director
Research Services Division

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify AFGL/DAA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFGL-TR-85-0169	2. GOVT ACCESSION NO. AD-A162 134	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* MASK.DOC		5. TYPE OF REPORT & PERIOD COVERED Scientific Report No. 1
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. Stevens		8. CONTRACT OR GRANT NUMBER(s) F19628-83-C-0090
9. PERFORMING ORGANIZATION NAME AND ADDRESS BEDFORD RESEARCH Associates 4 De Angelo Drive Bedford, MA 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62101F 9993XXAI
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Geophysics Laboratory Hanscom AFB, MA 01731 Paul Tsipouras, AFGL/SIA		12. REPORT DATE August 1985
		13. NUMBER OF PAGES 38
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19 KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

MASK
CIF (Caltech Intermediate Form)
Cornell Format
Electromask Code

20 ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

MASK.DOC is a description of the MASK Programming System for designing microwave integrated circuits. It is divided into six major parts:

1) Part 1 is an introduction to the MASK program. It describes briefly the function of all the modules and related programs of MASK;
2) Part 2 describes how a user goes about creating a MASK layout, converting it to CIF, and translating it to either Cornell or Electromask code. It also explains how to transfer the code to a magnetic tape and how to verify the tape.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Block 20 (continued from previous page)

3) This section explains how to add a new MASK structure to the structure. It tells which modules to change and how to go about it;
4) This is a programmer's description of MASK. It explains the function of each subroutine in all modules and related programs;
5) This section informs the user how to compile and link all the modules and related programs so as to ready the programs for execution; and
6) Special structures are described here. This section describes the structures that are limited to translation to either Electromask or Cornell format but not both.

TABLE OF CONTENTS

DTIC
ELECTE
DEC 9 1985
S D
B

QUALITY INSPECTED 3

A-1

MASK.DOC

MASK.DOC is a description of the Mask Programming System and its related programs and files. It is informative to both the user and the programmer.

1. INTRODUCTION TO THE MASK PROGRAMMING SYSTEM

A. MASK3xx - MASK3xx is the main module of the Mask Programming System. It contains all the general subroutines used in creating a MASK layout. It also controls the activity of the three other modules (CIF3xx, SERV3xx, and COLR3xx) which make up the MASK system. This module is described in detail in the programmer's section (Section 4).

B. COLR3xx - This module contains all the structure display routines. Each structure has its own display routine. The structures are all displayed in color using a Tektronix 4100 series terminal. This module is invoked whenever a structure is displayed in the course of designing a MASK layout.

C. SERV3xx - This module contains all the structure service and user help routines. Each structure has its own service routine. These routines allow the user to create, connect, scale, change and move any available structure.

D. CIF3xx - This module contains all the Caltech Intermediate Code (CIF) generation routines. As in SERV3xx and COLR3xx, all structures have their own CIF routine. The command CIFILE in MASK invokes this module. Structure by structure, the entire source file containing the MASK layout is translated to CIF and placed into a file called SOURCE.CIF, where "SOURCE" is the name of the current MASK source file.

E. BW3xx - This module is the black and white version of COLR3xx.

F. EIRN3xx and CIRN3xx - These programs are translators from CIF to ELECTROMASK code and CIF to EBMF CORNELL format respectively. Both the ELECTROMASK and CORNELL format can be copied to tape and sent to a MASK manufacturer to fabricate the actual MASK. The input file to both programs is SOURCE.CIF, which is created by the MASK program. The output from EIRN3xx is SOURCE.EM, which is the ELECTROMASK code. The output from CIRN3xx is SOURCE.CR, which is the EBMF CORNELL format.

G. ECHK3xx and CCHK3xx - These two programs are tape verifiers. The ELECTROMASK code SOURCE.EM, created by EIRN3xx, or the CORNELL format file SOURCE.CR, created by CIRN3xx, is first copied to magnetic tape which will be sent to a MASK vendor. To verify that the data on the tape is correct, ECHK3xx or CCHK3xx is run. These programs take the contents of SOURCE.EM or SOURCE.CR, which are on tape, and translate them back to the MASK layout. The layout is then displayed to the screen flash by flash in black and white. This way you can check to see whether any of the flashes are incorrect.

H. TOTAPE.COM and TODISK.COM - These two command files copy a CORNELL or EMASK format file from disk to tape and tape to disk respectively.

I. CIFA3xx, COLRA3xx, SERVA3xx - These are continuations of the modules CIF3xx, COLR3xx, and SERV3xx respectively. They exist to provide faster compilation when debugging. 3xx is the version number and A, B... is the continuation number.

J. HELP3xx - This module contains the user help subroutines. The HELPME command in MASK invokes this routine and provides the user with a complete list and description of all MASK commands, operations, available structures, and related programs and files. The HELPME structure will also display the structure by levels.

K. OTHER FILES USED - Three files are used with the MASK program:

MESSAGE.TXT - This file contains messages used with MASK and ETRN3xx.

HELPMES.TXT - This file contains all the help information used by the command HELPME in MASK.

LOOKUP.TAB - This file contains the flash table for numbers and letters to be entered into a CIF file.

2. HOW TO GENERATE AND CHECK A MASK TAPE

A. RUN MASK - The first step in generating a MASK tape is to design the MASK using the graphics layout program. Structure by structure, the MASK can be designed on the Tektronix 4100 series color graphics terminal. Once the MASK is created, it must be translated to Caltech Intermediate Form (CIF). This is done with the command, CIFILE. This translates the MASK layout into CIF and places it into a file called SOURCE.CIF, where source is the name of the current source file.

B. RUN ETERN3xx or CIRN3xx - Once the CIF file is created, it is time to choose which data translation you want. If you desire CIF to elecktromask code (EMASK), run program ETRN3xx. If you choose CIF to EBMF CORNELL format, run program CIRN3xx.

ETRN3xx - The only input file needed for ETRN3xx is the CIF file called SOURCE.CIF. The program will prompt you for the name of the CIF file. The output files will automatically be the defaults. The first one is the user-readable EMASK code file (default is SOURCE.ER), and the second one is the machine-readable EMASK file (default is the SOURCE.EM). Once these files are opened, the program will prompt you on which levels of the mask you would like to omit from translation to EMASK. Enter the levels and then "99" to start execution. The program will display information such as the mask size and number of flashes per level.

When the program terminates, files SOURCE.ER and SOURCE.EM are entered into your default directory. "SOURCE" is the current source file.

CIRN3xx - The input file for CIRN3xx is also SOURCE.CIF. The output file is automatically opened as SOURCE.CR. This file will contain the CORNELL format. As in EIRN3xx, the program will prompt you for levels to be omitted from translation. E nter the levels and then "99" to start execution. The file SOURCE.CR is entered into your default directory as the program terminates.

Now that SOURCE.EM or SOURCE.CR is created on disk, it must be copied to magnetic tape, which will be sent to a mask vendor.

Instructions for copying an EMASK or CORNELL file to tape: First execute the command file TOTAPE.COM. The program will prompt you to mount and load the tape physically on the tape drive. Then it will prompt you for the file to be copied. The .EM extension will automatically tell the program it is an EMASK file while a .CR will identify it as a CORNELL file. The file is copied to tape; then it is rewound and turned off-line for dismounting.

C. ECHK3xx and CCHK3xx - Now that the EMASK or CORNELL files are on tape, they must be verified before they are sent to the vendors. The verification is done using the program ECHK3xx for checking SOURCE.EM, or CCHK3xx for checking SOURCE.CR. First, the files have to be copied back to disk. This is done by running the command file TODISK.COM. The program will prompt you to mount and load the tape which has the CORNELL or EMASK file on it. It will then prompt you for the file name of the file to be created on disk. This will be either SOURCE.EC or SOURCE.CC. The extension will automatically signal the program whether the file is EMASK or CORNELL format. The file is copied to disk; then it is rewound and turned off-line for dismounting.

ECHK3xx - The input file is the EMASK file copied off the tape called SOURCE.EC. The program will prompt whether you want to verify the mask level by level, clearing the screen in between, or to display all levels of the mask on one screen. The program will then begin displaying the mask flash by flash. When the display is finished, the program prompts if you want to scale the display to check areas that appear small on the screen. Select a scale factor and the screen will expand the area chosen by the cross-hairs. You can go back and forth between the plot and expand menus until you have plotted and verified all the levels you wanted. The program will also prompt you to send the screen to the copier after each display. To terminate the program, choose "99" in the plot menu.

CCHK3xx - CHCK3xx takes SOURCE.CC, the CORNELL format file, from tape as input. The program also starts by prompting the user for levels to be displayed on the screen. Like ECHK3xx, this program will display the mask layout flash by flash for the chosen levels. The display is automatically scaled so the mask will use the entire screen for clarity. A user scale is also available. After each display, you are prompted for the option of a hardcopy and then sent back to the Levels Menu. Once you have display, checked or copied all the levels you desire, enter "99" in the Levels Menu to exit the program.

3. HOW TO ADD A NEW STRUCTURE TO THE LIBRARY

To add a new structure to the library, four modules have to be updated:

A. MASK3xx - The only addition to MASK3xx is in the function ISTCOD. ISTCOD takes the character name of a structure and codes it into an integer. For example, ISTCOD codes the structure MICROS, a microstrip, as Structure 1. The programmer must enter the new structure name into the array of strings called NAME. This is

done in the Data Statement. The structure name must be 6 characters long. Remember to change the number of blank strings in the Data Statement. The placement of the string in array NAME determines its structure number. For example, the structure BONDPD, a bond pad, is the 23rd string in NAME, thus its structure code is 23. That is all for MASK3xx.

B. SERV3xx - All structures are created and serviced in this module. The easiest way to add a new SERVE routine is to search for an existing SERVE routine of a structure with a similar design, then copy this routine and change the necessary parameters to fit the new structure. It is good to have printout to look at. For faster compilation while debugging the new structure, the new service routine should be written into one of the smaller service continuation files, SERVA3xx.

Start by changing the name to SERVxx, where xx is the integer structure code defined in ISTCOD. The parameters passed to this subroutine must be in the following order:

```
(IFLAG,CNNODX,CNNODY,ZETA,NODE,NSQINQ,NCHAIN)
```

All these parameters are used by this routine. If the structure is multi-level, then the level array LAYS(NLAY) must be defined. NLAY is the number of levels in the structure. Next come the parameter and common blocks. The following must be present:

```
PARAMETER(ISRC=3, IXCODE=xx)
PARAMETER(NTYPES=64,NEACHT=99)
PARAMETER(NUMVCS=200,NIWO=2)
COMMON /INDEXT/ LAYTAB(NTYPES,NEACHT)
COMMON /POLYV/ V(NUMVCS,NIWO)
```

The format statements are next. They should contain all the prompts for the creation of the structure. All structures have a placement X, placement Y, and an angle prompt. The remaining format statements should be prompts for the important size parameters of the structure. The last two formats (usually 95 or 96) are used with the LISTEM command and will be described later.

The next statement is an implicit GO TO statement using IFLAG. IFLAG is a variable passed to this routine which decides what operation to perform on the structure. IFLAG can be an integer from 1 to 10. The GO TO statement is as follows:

```
GO TO (100,200,300,400,500,600,700,800,900,1000) IFLAG
CALL MESSAGE(1)    !BAD IFLAG, END ROUTINE
GO TO 90000
```

Now the creation of the structure begins.

At label 100 (IFLAG=1), the structure angle is prompted and read in.

At label 200 (IFLAG=2), the registration point RX,RY is prompted and read in. IFLAG is then checked if equal to 2. If yes, then set angle theta = 0.0.

At label 300 (IFLAG=3), the rest of the structure's parameters are prompted and read in. The subroutines AREAD and IREAD should be used for reading in reals and integers, since defaults can be specified. Next, the layers of the strucutre have to be set. For a single layer structure, just a prompt and an IREAD call are needed. For a multi-layer strucutre, subroutine LYRSET must be called, passing the array LAYS and the number of layers. This routine will return the layer numbers for all layers of the structure in the array LAYS. Now set ICHAIN (the structure's chain number) to 0.

Now check if IFLAG=3. If it does, then this structure is being connected to the node of another structure. First set ICHAIN = NCHAIN. NCHAIN is the chain number of the structure being connected to. The angle and registration point of the new structure must be calculated relative to the position of the connect node CNNODX,CNNODY. ZETA is the angle of the connect structure. For example, for the varactor diode, structure number 15, the registration point is calculated by:

```
IF(NODE .EQ. 1) THEN
   THETA = ZETA - 180.
   RX = CNNODX + (0.7*WMESA)*SIND(ZETA-180.)
   RY = CNNODY - (0.7*WMESA)*COSD(ZETA-180.)
ELSEIF(NODE .EQ. 2) THEN
   THETA = ZETA
   RX = CNNODX - (W3 + 2.8*ELGATE)*SIND(ZETA-180.)
   RY = CNNODY - (W3 + 2.8*ELGATE)*COSD(ZETA-180.)
ENDIF
```

This is all calculated using geometry. It is easiest done by drawing a sketch of the structure connected to another and finding the X and Y distances to the registration point from the connect node. Once this is done using all the structure's nodes as reference, the IFLAG=3 condition ends. Now the following three subroutine calls are made:

```
CALL GETIDC(IXCODE,NSEQ,INDEXF)
CALL CPOSIT(INDEXF,ICHAIN)
CALL CLRDAR(ISRC,INDEXF)
```

GETIDC takes the structure code IXCODE, assigns it a sequence number (NSEQ), and assigns an index number (INDEXF) to the source file. CPOSIT assigns the structure a chain number and returns it in ICHAIN. CLRDAR clears the source file record, INDEXF, where the structure's parameters will be stored. ISRC is the logical

unit number for the source file. The next move is to write the parameters to the source file. The first six parameters must be in the following order, followed by the layers and the size parameters:

```
WRITE(ISRC, REC=INDEXF) IXCODE,NSEQ,RX,RY,THETA,ICHAIN,
     LAYS(1),...,LAYS(NLAY),PARAM1,...,PARAMn
```

To display the newly created structure, call DISPONE(INDEXF,ISRC). GO TO 90000 will end creation of the structure.

At label 400 (IFLAG=4) designates that this structure is the one being connected to. Therefore, the X,Y value of the connect node must be calculated. First, the parameters of this structure have to be read from the source file as follows:

```
READ(ISRC, LAYTAB(IXCODE,NSQINQ)) NSTR,NSEQ,RX,RY,THETA,ICHAIN,
     LAYS(1),...,LAYS(NLAY),PARAM1,...,PARAMn
```

The array LAYTAB is the layout table of the entire mask. Each structure is entered into LAYTAB when created. The two subscripts IXCODE and NSQINQ will located the record number of the structure in the LAYOUT table.

In this section, the X,Y location of each node is calculated relative to the registration point. Also, the normal angel (ZETA) of the structure is calcualted at each node. For the varactor diode, the calculation is as follows:

```
IF(NODE .EQ. 1)THEN
   ZETA = THETA + 180.
   V(1,1) = 0.0
   V(1,2) = 0.7*WMESA
ELSEIF(NODE .EQ. 2)THEN
   ZETA = THETA + 180.
   V(1,1) = 0.0
   V(1,2) = -W3 - 2.8*ELGATE
ENDIF
```

The array V is the vertice array where the first subscript is the vertice number and the second subscript is 1 for the X coordinate, and 2 for the Y coordinate. For example, V(1,1) is the X coordinate of the first vertice, and V(3,2) is the Y coordinate of the third vertice. Now the vertices of the nodes have to be transformed to virtual coordinates using subroutine XFORM.

```
CALL XFORM(RX,RY,THETA,1)
CNNODX = V(1,1)
CNNODY = V(1,2)
GO TO 90000
```

The 1 in XFORM means only one vertice ina rray V is transformed. CNNODX,CNNODY are the coordinates of the connect node n question. IFLAG=4 is completed.

At label 500 and 600 (IFLAG=5 and 6) are not currently used. CALL MESSAGE(1) and GO TO 90000 should be written at both these labels.

At label 700, the new structure is allowed to be scaled. First, MESSAGE(56) is called. This prompts the user to enter a scale factor for each scalable parameter. Now read in the parameters using LAYTAB identical to the READ at label 400. The next step is to prompt the user and read in each scale factor for each parameter. Use AREAD to read in the factor giving 1.0 (NO SCALE) as a default. Next, check if any factors were 0 by multiplying them together and testing for 0 result. Now scale the old parameters by multiplying them with the scale factors. Next, call CLRDAR using LAYTAB to clear the old record.

```
CALL CLRDAR(ISRC,LAYTAB(IXCODE,NSQINQ))
```

Then write the new scaled parameters back to the source file. End the scale with GO TO 90000.

At label 800, the structure is changed. First, call MESSAGE(19), then read in the parameters from the source file using LAYTAB as at label 400. Now prompt for all changeable parameters using AREAD and IREAD to read in the new ones. Use the old ones as default. This includes RX, RY, and THETA. Now change the layers using subroutine LYRSET and array LAYS for multi-layers. Call CLRDAR using LAYTAB as at label 700 and write the new parameters back to the source file.

At label 900, the LISTEM command is invoked. The important parameters of the structure are printed either to the screen or to a list file called LISTEM.LST. Start by reading in the parameters using LAYTAB. Then check the passed parameter NCHAIN. If NCHAIN = 1, then write to a file using the passed parameter NODE as the logical unit number. Otherwise, print to the screen. The parameters are printed according to a format (either 95 or 96). Format 95, which prints to a file, prints all parameters on one line. Format 96, which prints to the screen, forms a new line after every 80 characters. This is so all parameters fit on the screen. The important parameters include the character name of the structure, then NSEQ,RX,RY,ICHAIN, all layers and all size parameters. For the varactor diode, LISTEM is as follows:

```
IF(NCHAIN .EQ. 1)THEN
   WRITE(NODE, 95) NSEQ,RX,RY,LAYS(1),LAYS(2),LAYS(3),
                   ICHAIN,ELGATE,DGATE,ELOHM,WMESA,W3
ELSE
   WRITE(*, 96) NSEQ,RX,RY,LAYS(1),LAYS(2),LAYS(3),
                ICHAIN,ELGATE,DGATE,ELOHM,WMESA,W3
ENDIF
```

End LISTEM with GO TO 90000.

Label 1000 (IFLAG=10) is used to change an existing structure's X,Y location. All that is done here is that the old parameters are read out using LAYTAB. The record is cleared by a call to CLRDAR. and the parameters are read back in substitutting CNNODX,CNNODY for RX,RY. Label 90000, return, and end finish of the service routine. Now go to the first routine of this module called SERVER. In SERVER, add the condition for ICODE=xx where xx is the structure number of the new structure. The code will be as follows:

```
ELSEIF(ICODE .EQ xx)THEN
    CALL SERVxx(IFL,X,Y,A,NODE,NSEQ,NCHN)
```

Make sure all the parameters present are passed.

C. COLR3xx - This module contains all the structure display routines. Like the SERVE routines, the DISPLAY routines are identified by their structure number, and each structure has its own. To add a new structure display routine, first copy one from an existing structure of similar design. A hardcopy to look at is helpful. For faster compilation while debugging the new structure, the new display routine should be written into one of the smaller display continuation files, COLRA3xx.

First change the name to DISPxx, where xx is the structure number. The passed parameters should be INDEX and IFILE. Make sure the following parameter and common blocks are present:

```
PARAMETER(NUMVCS=200, ITWO=2)
PARAMETER(NLYRS=14)
COMMON /PLOYV/ V(NUMVCS,ITWO)
COMMON /RELAYS/ LAYERS(0:NLYRS)
```

Next, read the parameters from the source file using IFILE as the logical unit number and INDEX as the record number.

Now the display is set up layer by layer. A layer will be displayed only if it is set to 1 in the LAYERS array. For each layer, LAYN, the value of LAYERS(LAYN) is tested for 1. If yes, then display; if ELSE, do not. Also, LAYERS(0) is the universal display setting. Sometimes a structure does not display well when all the layers are shown simultaneously. It is up to the programmer to set which layers are to be displayed when LAYERS(0) is 1.

To display a layer, you must fill the vertice array V with X,Y coordinates of an enclosed polygon. This is best done by looking at a sketch of the strucutre and, one by one, fill the array V with consecutive points of an enclosed polygon. Coordinates are found relative to the registration point. The polygon must be enclosed in order to fill with color. Once array V is filled with the points of the polygon, XFORM must be called to translate the points to screen coordinates.

```
CALL XFORM(RX,RY,THETA,NPTS)
```

NPTS is the number of vertices in the defined polygon. To fill with the right layer color, call the following:

```
CALL LCOLOR(LAYN,NUMCOL)
CALL SELPAT(NUMCOL)
```

LCOLOR takes in the layer number and returns the proper integer color number for that layer. SELPAT then selects that color for a fill pattern. A call to BOUND draws the polygon on the screen and fills it with color.

```
CALL BOUND(VFIRST,VLAST)
```

VFIRST and VLAST are the first and last vertice of the polygon. Usually, the structure cannot be defined by just one polygon. For every polygon ,XFORM,LCOLOR,SELPAT, and BOUND must be called. When one layer is finished, go on to the next layer in the structure. Once all polygons for all layers are designed, the subroutine ends.

Go back to the first subroutine in this module, called DISPLAY. Enter the condition to call the new structure display routine.

```
ELSEIF(NSTR .EQ. xx) THEN
    CALL DISPxx(LAYTAB(NTSR,NSEQ), ISRC)
```

Next go to subroutine DISPONE and add:

```
    ELSE
+   IF(NSTR .EQ. xx) THEN
      CALL DISPxx(INDEX, ISRC)
```

This concludes addition to COLR3xx.

D. CIF3xx - This module contains all the routines for CIF generation. Again, copy a CIF routine of an existing structure with similar design. For faster compilation while debugging the new structure, the new CIF routine should be written into one of the smaller CIF continuation files, CIFA3xx. Change the name of the routine to CIFxx. The parameters passed to this routine must be, in order, INDEX, LEVNUM, GEOMOP. The following parameter and common blocks must be present:

```
LOGICAL GEOMOP
PARAMETER(ISRC=3,ICIF=16)
PARAMETER(NUMVCS=20,ITWO=2)
COMMON /POLYV/ V(NUMVCS,ITWO)
```

Next, a statement function converting meters to hundredths of microns:

```
ICON(TERM) = INT(TERM/(0.00000001) + 0.5)
```

Read the parameters of the structure from the source file using ISRC as the logical unit number and INDEX as the record number. The current level being processed is passed to the routine through LEVNUM. The next statement should check if any of the layers of this structure equals LEVNUM. If yes, then set GEOMOP = true. This tells the main CIF routine that geometric output has occurred. If no output, then go to the end of the routine.

To produce geometric output (translate to CIF), first derive a directional vector using structure angle THETA:

```
IDIRX = INT(COSD(THETA)*1.E-5)
IDIRY = INT(SIND(THETA)*1.E-5)
```

Test each layer separately if it equals LEVNUM. If yes, then translate only that layer.

Each layer must be broken up into CIF rectangles. The width, length and midpoint of all rectangles must be calculated. The midpoint coordinates should be calculated relative to the registration point and entered into the vertice array V. Then XFORM is called to transform them to screen coordinates. For example, we have a rectangle with width = 0.01, length = 0.02, and midpoint at X,Y. The following statements occur:

```
WID = 0.01
LEN = 0.02
V(1,1) = X
V(1,2) = Y
CALL XFORM(RX,RY,THETA,1)
```

Next, a call to CIFBOX is made. CIFBOX takes the rectangle's parameters and places them in a CIF file. For the above data, the call is as follows:

```
CALL CIFBOX(ICON(LEN), ICON(WID), ICON(V(1,1)), ICON(V(1,2)),
            IDIRX, IDIRY)
```

If a polygon is desired, the routine POLYGN is called. The polygon has to be four-sided, and the four points must be defined in array V. The call is as follows:

```
CALL POLYGN(ICON(V(1,1)), ICON(V(1,2)), ICON(V(2,1)),
            ICON(V(2,2)), ICON(V(3,1)), ICON(V(3,2)),
            ICON(V(4,1)), ICON(V(4,2)),
```

Notice that the parameters are converted to hundredths of microns using ICON. Once all rectangles and polygons for all layers are done this way, the subroutine ends.

Go to subroutine CIFER. Add the following condition:

```
ELSEIF( NSTRUC .EQ xx)THEN
    CALL CIFxx(IDXSRC, LEVNUM, GEOMOP)
```

Addition to CIF3xx is complete. This also completes the addition of a new structure to the library.

4. PROGRAMMER'S DESCRIPTION OF THE MASK PROGRAM

A. MASK3xx - This is the main module for the MASK system and contains all the general-purpose subroutines.

Subroutine MAIN - This is the backbone of program MASK. It opens all necessary files and does most of the initialization.

At label 10070, the LAYERS array is initialized to 0. Then the chain file, CHAINS.DAT, and the message file, MESSAGE.TXT, are opened.

At label 10080, the layout table, LAYTAB, is set to 0.

At label 10090, the GRAPICS is initialized by calls to CODETEK and INITT. Then the screen is cleared and the banner printed. Subroutine SRCOPEN is called, which opens the source file OLD or NEW. If ESCAPE was entered, the program terminates.

If the source file was opened NEW, the mask size parameters are prompted and read in and placed in the first record of the source file. If the source file is OLD, a backup copy is made and the chains and layout table are built.

Next, the screen coordinates are set with calls to DWINDO and CVSCAL. The program now jumps to label 20000 where the mask prompt is printed and waits for a user command. The next statement reads in the user command and sends it to subroutine PARSER. PARSER processes the command and returns. If logical variable IMDONE is true, then the program closes all files and terminates. Otherwise, the mask prompt is printed and the program waits for a new command.

Subroutine PARSER - This routine parses the first command (6 characters) in the mask command line entered by the user. It then calls the appropriate routines to process that command. The commands are parsed, in order, as follows:

ESCAPE	- Terminate program.
HELPME	- If HELPME XY, then, using the crosshairs and subroutine VCURSR, find any X,Y screen location.
SIZEIT, MASIZE	- Both print out the mask dimensions to the screen.

EXPAND	- With the use of VCURSR and the crosshairs or user-entered coordinates, scale an area of the screen. The scaling is done by subroutine CVSCAL.
PATGEN	- Not available.
ELBEAM	- Not available.
LAYOUT	- Displays the existing layout by calling subroutine DISPLAY.
FILEIT	- Calls subroutine SRCFILE which allows the user to save the current mask layout without exiting program.
MOVEST	- Move a structure by calling SERVER with IFLAG = 10. The distance X,Y to move is prompted to the user.
BLOKMV	- Same as MOVEST except the entire chain is moved.
HELPST	- Not available.
DELETE	- Delete an entire chain of structures with a call to DELCHN.
DUPLIC	- Duplicate a chain of structures with a call to DUPCHN.
SLAYER	- Sets levels of the existing mask layout for display. This is done in subroutine LYMENU.
CIFGEN	- Processes the mask command, CIFILE. Call GENCIF to generate the mask layout to CIF.
LISTEM	- List the contents of the source file. This is done by calling SERVER with IFLAG = 9 for all structures in the source file.
LAYTAB	- Print the layout table to the screen.
GETSUP	- Implement the super structure utility. Call SUPER1.

If the command was a 6-character structure name, ISTCOD is called to get the structure code and PARSE2 is called to parse the structure operations. PARSE2 returns IFLAG which represents the type of structure operation to be performed.

At label 10000, IFLAG=1, structure connections are performed. The connection commands are ++,H+,V+,A+. These operations are done in subroutine COMPOS.

At label 20000, IFLAG=2, a new structure is placed at location X,Y or at angle THETA (AA). This is accomplished with a call to SERVER with IFLAG=1 and 2 respectively.

At label 30000, IFLAG=3, structure move operations are done (UP,DN,RT,LF). This is done by calls to MOVACH.

At label 40000, IFLAG=4, structure scale (SC), Change (CH), and delete (—) operations are performed. Call SERVER, IFLAG=7, to scale, call SERVER, IFLAG=8, to change, and call DELSTR to delete a single structure.

At 90000 the routine ends.

Subroutine PARSE2 - This routine parses the command line for structure operations and returns the desired nodes and sequence numbers.

Subroutine COMPOS - This routine does all structure connect operations. First, it calls SERVER with IFLAG=4 to get the existing structure's connect node coordinates. Then, in order, the code for A+,H+,++, and V+ are listed. The new strctures being connected are created with a called to SERVER, IFLAG=3.

Below is a list of the rest of the subroutines in MASK3xx.

NEGVEX - Determines if two structure angles are connect-compatible.

FUNCTION ISTCOD- Encodes a 6-character structure name to an integer code. All structures are entered in the array name and assigned an integer according to placement.

FUNCTION ISTSEL- Used for selecting choices of a menu.

CVSCAL - Scales user space by the specified scale factor. It uses Plot-10 routine DWINDO.

XFORM - Translates the points in the vertice array V relative to the registration point, X,Y, and angle THETA, of a structure.

GETIDC - This routine enters a new structure into the layout table, assigns it a sequence number and an index to the source file.

STRIDEN - Gets the structure code and sequence number of an existing structure.

DELCHN - Deletes a chain of structures. It calls DELSTR.

DELSTR - Deletes one structure.

CPOSIT - Deposits a new structure into a chain.

LYMENU - Sets the layer display array.

DUPCHN - Duplicate a chain of structures.

MOVACH - Move a chain of structures. It calls SERVER, IFLAG=10.

XLATHZ, XLATVR - Translates horizontally or vertically all structures created after a given structure.

LOGICAL FUNCTION
MEMBER - Determines if an operator is a member of a given set.

SKIPBL - Skip all preceding blanks in a string.

LOGICAL FUNCTION
ASKQIT - Prompts user to quit or not.

SREAD - Read in a string with the option of a given default.

LOGICAL FUNCTION

NAMEOK - Checks if a command or name is 6 characters starting with a letter and the rest alphanumeric.

IN2CHR - Returns the integer value of a 2-digit decimal number.

INTCHR - Returns the integer value of a numeric character.

SRCFILE - Allows manipulations to the source file. The command FILEIT uses it.

ESCAPE - Allows you to enter the name of the exit file and copy the working source to it when exiting MASK.

BACKUP - Creates a backup of an 'old' source file to use as a working copy.

SRCOPEN - This routine prompts the user to open an old or new source file.

SUPER1, SUPER2,
SUPER3, SUPER4,
SUPERD - These routines get a super structure from an existing source file and add it to the current layout.

MESSAGE - Reads from the message file, MESSAGE.TXT, the proper MASK message and prints it to the screen.

IREAD,AREAD - Read in integers and reads with the option of a given default.

CORNER - Not used.

ASKYNO - Prompts the user for a Yes or No response.

ATTEN - Halt the program until a key is struck.

LOADVX - Load array V with proper position vectors.

LKUPOS - Read in the flash position for inputted character.

CODETEK - Changes the setting of the TEKTRONIX 4107 terminal to graphics mode.

CODEDIT - Changes the setting of the TEKTRONIX 4107 terminal to edit mode.

This concludes module MASK3xx.

B. SERV3xx and SERVA3xx,SERVB3xx - This module contains all the structure service routines.

Subroutine SERVER - This is the driver module for all structure service routines. It directs the flow of the program to the proper structure routine.

Subroutines SERVØ1 through SERVnn - These are all the individual structure service routines. All existing structures have their own. To add or modify one of these routines, go to Part B, Section 3, on how to add a new structure to the library.

This completes module SERV3xx.

C. COLR3xx and COLRA3xx, COLRB3xx - This routine contains all the structure display routines.

Subroutine DSPLAY reads the current source file and displays each structure one by one by calling the proper structure display routine. Each structure has its own routine.

Subroutines DISPER, DISPONE display only a single inputted structure.

Subroutines DISPØ1 Through DISPnn are individual structure display routines. They ae described in detail in Part C, Section 3 (how to add a new structure to the library).

BOUND - Draw the boundary of an enclosed polygon defined in the vertice array V. Also the polygon is filled with the right color. Plot - 1Ø routines are used.

LYRSET - Sets the MASK layers for a multilayered structure.

CLRDAR - Clears a record of the source file.

LCOLOR - Assigns the proper color number for the proper layer.

This completes module COLR3xx.

D. CIF3xx and CIFA3xx, CIFB3xx - This module contains all the structure CIF generation routines.

GENCIF - This is the driver routine for the generation of CIF. It opens the CIF file, SOURCE.CIF then structure by structure, it reads from the source file and calls the appropriate CIF generation routine.

Subroutine CHOGEN prompts the user for type of CIF generation. The user has the choice of CIF generation for EMASK or CORNELL format translation.

Subroutines CGENID, CGENID2, CGENID3 generate CIF code for a MASK ID.

Subroutines CFNEXT, CIFGET, CFSKBL, SHNAME, CFPAS1, ITALBN, CIFSRT, CIFSRT2 and CIFSRT3 are currently not used.

Subroutine CIFLIM enters the MASK limits to the CIF file.

Subroutine CIFCOM enters comments into the CIF file.

Subroutine CIFBOX enters a CIF rectangle into the CIF file.

Subroutine POLYGN enters a CIF four sided polygon.

Subroutine CIFLAY enters a change of layer into the CIF file.

Subroutine CIFER is the driver to all individual structure CIF routines.

Subroutines CIF01 through CIFnn are the individual structure CIF generation routines. They are described in detail in Part D, Section 3.

Subroutine FRAME draws a frame around the MASK layout.

Subroutine DRWNUM draws a number on the display (not used).

This completes module CIF3xx.

E. EIRN3xx - This program translates CIF code to ELECTROMASK code.

Subroutine MAIN opens is the MESSAGE file, MESSAGE.TXT, and prints the banner. GENPAT is then called.

Subroutine GENPAT prompts and opens the CIF file at Label 1000. At Label 1100, the machine readable EMASK file and the user readable EMASK files are opened. PRSCIF is called next. This is where the translation takes place. When the translation is completed, CVRETP is called to create the machine readable file. This ends the routine.

Subroutine PRSCIF parses the CIF file and translates it to EMASK code. At label 450, the CIF file is read to find all the layer numbers available. At label 500, layers are printed and the user is prompted to enter the layers not to be translated to EMASK. At 10000, the parsing begins by reading a line of CIF code. If the line is a CIF box, then the dimensions are converted to tenths of microns and sent to subroutine EMBOX which creates an ELECTROMASK box. If the CIF line is a layer, then EMLAYR is called to process an EMASK layer. If the line is an end statement, then EMEND is called. If the line is a comment, then MASK size parameters are read in and EMWRLN is called. This writes the comment to the EMASK file.

Subroutine PREFIL prefills an 800 character array called BLOCK, with a beginning and end of block character. This array is used in creating the machine readable EMASK file which is written in 800 character blocks.

Subroutine FLUSH prints the entire 800 character array, block, to the user readable EMASK file. Then it calls PREFIL.

Subroutine CVRETP converts the user readable EMASK file to the 800 character record machine EMASK file.

Subroutine EMWRIT writes an EMASK operator and its parameter to the array block.

Subroutine EMWRLN writes a CIF comment into the array block. The comment is enclosed in quotes.

Subroutine CH6INT converts an integer to a six (6) character numeric string.

Subroutine EMLAYR processes a CIF layer command. FLUSH is called to empty the array block to the EMASK file. That terminates the previous layer. The new layer is started by printing the level number.

Subroutine EMBOX takes the dimensions of a CIF rectangle and breaks them down to flashes.

Subroutine EMFLSH writes out a flash to the EMASK file. The parameters that describe the flash are W,V,U,X,Y which are width, height, angle, X location and Y location.

Subroutine MESSAGE prints the proper message from MESSAGE.TXT.

Function ASKQIT prompts user to quit.

Subroutine SREAD reads a string with optional default.

Subroutine SKIPBL skips preceding blanks of a string.

Function SHNAME extracts a short name from a string.

This concludes ETRN3xx.

F. ECHK3xx - This program verifies the machine readable ELECTROMASK code written to tape.

Subroutine ELCHECK starts by printing the banner at label 25 after all the format statements. Next the EMASK file is opened. Then the first 800 character record is read to find the MASK size parameters. With calls to PLOT - 10 routines INITT and TWINDO, the screen size of the Tektronix 4107 is set.

At Label 140, the layers menu is printed. The user can choose if he wants all layers of the MASK to be verified separately or all on one screen.

At Label 190, the user is prompted if a copy of the screen is wanted. Then the expansion menu is printed. The user can expand any portion of the screen by a given scale factor. The scaling is done in routine SCALE.

At Label 200, PARSEM is called. PARSEM parses the EMASK file and displays the layout flash by flash.

At Label 9100, the routine ends.

Subroutine PARSEM parses the EMASK code. In order, it parses for the following EMASK operators: S(FLASH), Z(END OF LAYER), .(END OF BLOCK), "(COMMENT), "LEVEL"(NEW LAYER), U,V,W,X,Y.

Subroutine ISUBSTR reads a substring from a string.

Subroutine ERMROR prints EMASK error condition.

Subroutine EMRDTP reads EMASK file from tape to disk (not used).

Subroutine WRBLOC writes the 800 character array, block, to the screen (not used).

Subroutine BOX displays an EMASK flash to the screen around point X,Y.

Subroutine DASH is not used.

Subroutine FRAME draws a frame around MASK display.

Subroutine SCALE scales any portion of the graphics screen using VCURSR and DWINDO.

Subroutines SREAD and IREAD read strings and integers with the option of a given default.

This concludes ECHK3xx.

G. CIRN3xx - This program translates CIF code to EBMF CORNELL format.

Program MAIN - This routine contains the CIF parser.

At Label 350, the machine readable CORNELL file is opened. Then the CIFILE is prompted for and opened.

At Label 450, the CIF code is read through and the available layers are defined. These layers are printed at Label 500.

At Label 700, the user is prompted to enter which layers are to be omitted from translation from CIF to CORNELL format.

At Label 1000, the translation begins. The parser begins by checking for a CIF comment. The MASK size parameters are defined in these comments.

The parser then checks for a layer command. Subroutines ENDC and COMENT are called. These routines end the previous CORNELL layer and set up for the next one. Next, the parser processes a CIF End of File statement. Only ENDC is called.

The last condition checked is for a CIF rectangle. The parameters are read in and CORNELL format is computed. CORNELL format consists of the X,Y coordinate of the lower left point and the upper right point of the rectangle. This is easily computed from the CIF parameters. The CORNELL format is written to the file by the special format statement at label 400.

When the conversion is complete, all files are closed and the program terminates.

Subroutine COMENT initializes a new CORNELL format layer. This is done by writing the following two lines:

DE ROME AIR DEVELOPMENT CENTER
/CORNELL EBEAM LITHOGRAPHY SYSTEM

Subroutine ENDC terminates a CORNELL format layer. The string , END, is written to the file.

Subroutine SKIPBL skips preceding blanks in a string.

Subroutine SREAD reads a string with the option of a default.

This concludes CIRN3xx.

H. CCHK3xx - This program verifies a CORNELL format tape file.

Program MAIN - First the banner is printed and the CORNELL file is opened.

At Label 400, the MAX and MIN of the CORNELL parameters is defined. Then the graphics screen window is defined with a call to TWINDO and the virtual scale of the window is set with a call to DWINDO. This program automatically scales the window so the display covers the whole screen.

At Label 650, the number of levels available are printed to screen. The user is then prompted to enter which layer or layers to be verified.

At Label 1000, the CORNELL format is parsed. First a CORNELL layer is checked, then an end of layer command is checked.

No display is done until a CORNELL rectangle is found. Then the rectangle parameters are sent to subroutine BOUND which displays it on the screen. When the display is done, the user can exit or display different levels.

Subroutine SKIPBL skips preceding blanks in a string.

Subroutine SREAD reads a string with option of a given default.

Subroutine BOUND - Using PLOT-10 commands, a CORNELL rectangle is drawn to the screen.

Subroutine FRAME draws a frame around the MASK display.

This concludes CCHK3xx.

I. HELP3xx - This module contains the user HELP routines.

Subroutine HELPER is the driver routine for the MASK HELPME command.

Subroutines GIVHLP, HLPMSG, HLPRNT read and print the proper message from file HELPMES.TXT.

Subroutine HELPST displays a given structure by levels to the screen.

Subroutine INSYMB inserts a symbol in the display (not used).

Subroutine POINT draws a point on the screen.

5. COMPILING AND LINKING OF THE MASK SYSTEM

A. MASK.EXE - To create MASK.EXE, the following modules have to be compiled.

1. MASK3xx.FOR
2. COLR3xx.FOR
3. CIF3xx.FOR
4. SERV3xx.FOR
5. All the CIF, SERVE, and COLR continuation modules
6. HELP3xx.FOR

This is done using the FORTRAN statement as follows:

```
$  FORTRAN MASK3xx
$  FORTRAN CIF3xx
$  FORTRAN COLR3xx
$  FORTRAN SERV3xx
$  FORTRAN HELP3xx
$  FORTRAN CIFA3xx

        .
        .

$  FORTRAN CIFn3xx
$  FORTRAN COLRA3xx

        .
        .

$  FORTRAN COLRn3xx
$  FORTRAN SERVA3xx

        .
        .

$  FORTRAN SERVn3xx
```

The command FORTRAN can be abbreviated to just F. In compiling these FORTRAN modules, object files were created. (With EXTENSION .OBJ). these object files must be linked together to create the executable file (EXTENSION .EXE). This is done as follows:

```
$ LINK/EXE=MASK MASK3xx,COLR3xx,CIF3xx,SERV3xx,COLRA3xx, ...,
  COLRn3xx,CIFA3xx,...,CIFn3xx,SERVA3xx,...,SERVn3xx,HELP3xx
```

The executable file MASK.EXE now exists in the default directory. To run the program type:

```
$ RUN MASK
```

B. ETRN3xx and ECHK3xx - The two FORTRAN modules must be compiled to create the object files. This is done as follows:

```
$ FORTRAN ETRN3xx
$ FORTRAN ECHK3xx
```

To create the executable files, link both separately:

```
$ LINK ETRN3xx
$ LINK ECHK3xx
```

The executable files for both programs now exist in the default directory. To run the programs, simply type:

```
$ RUN ETRN3xx
$ RUN ECHK3xx
```

C. CTRN3xx and CCHK3xx - The two FORTRAN modules must be compiled to create the object files. This is done as follows:

```
$ FORTRAN CTRN3xx
$ FORTRAN CCHK3xx
```

To create the executable files, link both separately:

```
$ LINK CTRN3xx
$ LINK CCHK3xx
```

The executable files for both programs now exist in the default directory. To run the programs, simply type:

```
$ RUN CTRN3xx
$ RUN CCHK3xx
```

6. LIST OF STRUCTURES THAT CANNOT BE TRANSLATED TO BOTH EMASK AND CORNELL

A. Structures available for CORNELL translation only.

1. MESFET - TGATE with air bridge EPI.

B. Structures available for EMASK translation only.

1. MITERB - No miter and bend angle larger than 90.0 deg.

C. Structures available for EMASK and CORNELL translation but require separate code.

1. LANGEC - Lange coupler.
2. MITERB - With miter and bend angle at 90.0 deg.

This concludes the text file MASK.DOC.

END

FILMED

1-86

DTIC